BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

Black Codher Bootcamp

NÍ YÓ

BLACK CODHER

CODING PROGRAMME

UNIT 3
Javascript 102

CODING BLACK FEMALES

NÍ YÓ

RECAP

- Values
- Variables
- Functions

WHAT YOU'LL BE LEARNING TODAY?

BLACK CODHER
CODING PROGRAMME

- Statements
- Objects
- Operators
- Expressions
- Conditional Logic

BLACK CODHER

CODING PROGRAMME

STATEMENTS

STATEMENTS

By now you should've noticed that we've been using `;` in various places. A semicolon marks the end of a **statement**. So far we have been writing each statement on its own line, but this is not a requirement. There are a few exceptions, but you can add extra spaces and newlines wherever you like. The statements on the following page all mean the same thing.

STATEMENTS

```
console.log('Hello, how are you?');
```

```
console.log(
        'Hello, how are you?');
```

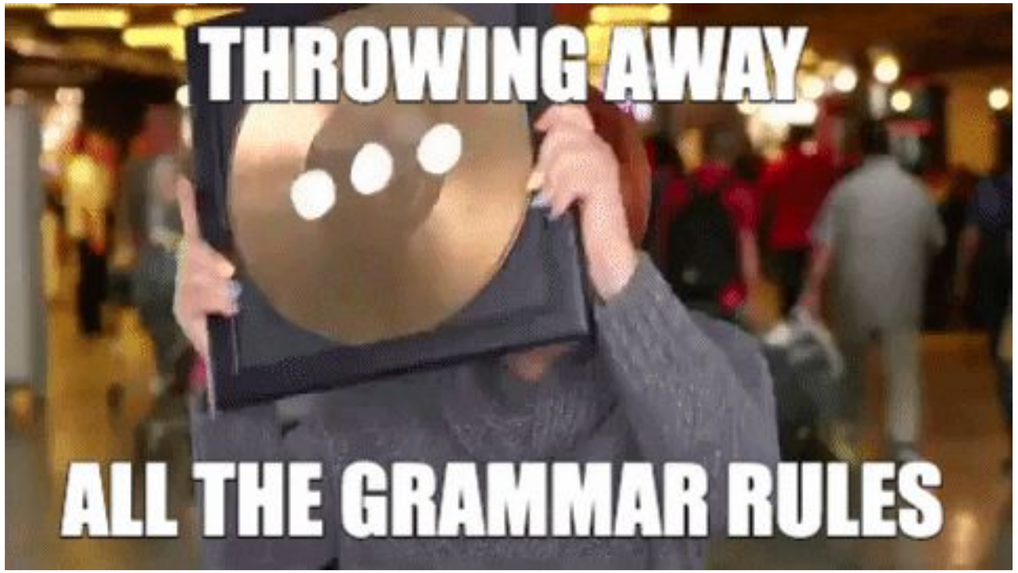
```
console.log(
    'Hello, how are
you?'
```

STATEMENTS

BLACK CODHER
CODING PROGRAMME

Use spaces and newlines to make your program easy to read. When you're just starting out, it won't be immediately clear to you what makes things easier to read, so just try to follow the patterns used in this course. As you read and write more programs this will start to make sense to you.

SEMICOLONS



Semicolons are needed after statements that do not end with a `}`. It is sometimes possible to leave out semicolons, however your program could break in a surprising way, so try to put them in the right places.

MULTIPLE LINES AND `return`

Be careful with the `return` keyword. You shouldn't add a newline immediately after the word return, because if you write this:

```
return
    a + b;
```

It is interpreted as:

```
return;
a + b;
```

MULTIPLE LINES AND `return`

If you want to break up a return into multiple lines, you can write it like this and it will work:

```
return (
    a + b
);
```

OBJECTS

Another type of value is an object. It can be created like this:

```
const person = {
    name: 'Monique',
    likes: 'pizza'
};
```

OBJECTS

The **object** referred to by the variable `person` has two **properties**:

- `name`
- `likes`

Each **property** can be accessed by writing `person.name` and `person.like`. This way accessing an object's **property** is known as **dot notation**

OBJECTS

A **property** is very similar to a **variable**, and can be the value stored can be changed using `=`:

```
person.likes = 'spaghetti';
```

REFERENCES

Objects are **values**, so it's common to say "the `person` **object**", but really it means "the **object** referred to by the `person` **variable**". It's important to understand this because several variables can refer to the same **object**.

TASK

```
const personA = {
    firstName: 'Monique',
    likes: 'pizza'
};
const personB = personA;

console.log("Before");
console.log(personA.name); console.log(person_b.first_name);

person_a.first_name = "Stacy";

console.log("After"); console.log(person_a.first_name); console.log(person_b.first_name);
```

Would you expect `personA.name` to be the same as `personB.name`?

METHODS

BLACK CODHER
CODING PROGRAMME

It is also possible to store a function in a property. When we do this, we use the word **method** instead of property. You call a method by combining the way you write properties with the way you write function calls:

```
console.log('Hello!');
```

You may recall this line from the start of the lesson. You now know enough to understand that `console` is an object, and `log` is a method on the console object.

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

BLACK CODHER

CODING PROGRAMME

VALUES

VALUES

Last session we learnt about **values** and here's a recap

Strings:

```
const name = "Black Codher";

console.log(name + ' is amazing!');
```

VALUES

BLACK CODHER
CODING PROGRAMME

Numbers:

```
const pi = 3.14;

console.log('The value of pi: ' + pi);
```

VALUES

Before moving on let's meet another important type: **booleans**.

A **boolean** can only be `true` or `false`.

Example:

```
const codeherIsAmazing = true;
const weatherIsGreat = false;

console.log('Is codeher amazing? ' + codeherIsAmazing);
console.log('Is the weather great? ' + weatherIsGreat);
```

BLACK CODHER

CODING PROGRAMME

OPERATORS

BASIC MATHS

Previously we covered the `+` operator. The other basic math operators are `-`, `*`, and `/`:

```
const x = 6;
const y = 3;
const addition = x + y;

console.log('Addition: x + y = ' + addition); // Addition: x + y = 9

const subtraction = x - y;

console.log('Subtraction: x - y = ' + subtraction); // Subtraction: x - y = 3

const multiplication = x * y;

console.log('Multiplication: x * y = ' + multiplication); // Multiplication: x * y = 18

const division = x / y;

console.log('Division: x / y = ' + division); // Division: x / y = 2
```

TASK

Try some other maths problem using the x and y variables?

OTHER MATHS OPERATORS

Other useful maths operators are `%`, `**`, `++` and `--`:

The **modulus** `%` operator returns the remainder when dividing one operand by another:

```
const x = 7;
const y = 3;
const modulus = x % y;

console.log('Remainder: x % y = ' + modulus); // modulus = 1
```

OTHER MATHS OPERATORS

The **exponentiation** `**` operator returns the result of raising the first operand to the power of the second:

```
const x = 7;
const y = 3;
const exponentiation = x ** y;

console.log('Exponentiation: x ** y = ' + exponentiation); // exponentiation = 343
```

OTHER MATHS OPERATORS

BLACK CODHER
CODING PROGRAMME

The **increment** `++` and **decrement** `--` operators return the result of adding one and subtracting one from an operand respectively.

```
let a = 7;
let b = 3;
const increment = a++;

console.log('Increment: a++ = ' + increment);

const decrement = b--;

console.log('Decrement: b-- = ' + decrement);
```

Notice that `let` is used to declare variables `a` and `b` because we are unable to modify a constant value.

COMPARISONS

The `===` operator compares two values, it returns the boolean `true` if they are equal and `false` if they are not.

```
const apples = 'apples';
const oranges = 'oranges';

const isEqual = apples === oranges;

console.log('Are apples and oranges the same? ' + isEqual); // false
```

COMPARISONS

The opposite of `===` is `!==`. It returns `true` if they are not equal, and `false` if they are.

```
const apples = 'apples'
const oranges = 'oranges'

const isEqual = apples !== oranges

console.log('Are apples and oranges the same? ' + isEqual) // true
```

You may also see `==` and `!=`, these are similar but have some quirks so it's generally recommended to avoid them. You can just use `===` and `!==` and they will always do what you expect.

GREATER OR LESS THAN

BLACK CODHER
CODING PROGRAMME

The > and < operators are *"greater than"* and *"less than"*. You can use them to tell which of two numbers is bigger.

```
const volunteers = 20;
const students = 24;
const pizzas = 25;

const moreStudents = students > volunteers;
console.log('Are there more students than volunteers?' + moreStudents);

const lessStudents = students < pizzas;
console.log('Are there fewer students than pizzas?' + lessStudents);
```

COMBINING OPERATORS

BLACK CODHER
CODING PROGRAMME

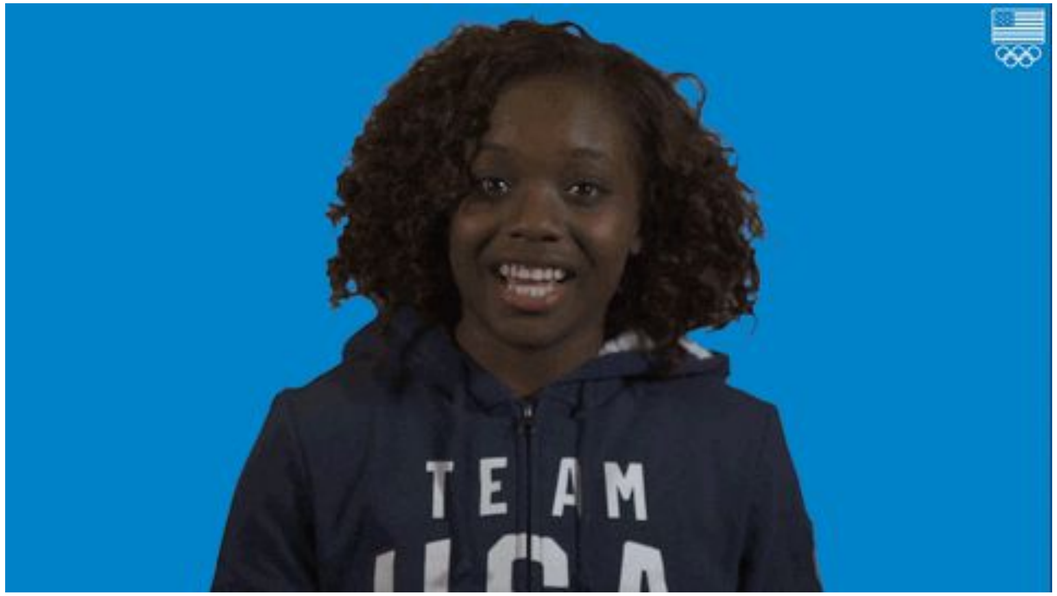
You can also combine operators:

```
const enoughPizzas = (coaches + students) < pizzas;

console.log('Do we have enough pizzas for everybody? ' + enoughPizzas);
```

TASK

Create 2 variables, one for your age and the other for the minimum driving age. Then do a `console.log` checking whether you are old enough to drive.

WHAT YOU'VE LEARNED SO FAR



- What a **boolean** is
- How to use **maths operators**
- How to compare **values**
- How to combine **operators**

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

BLACK CODHER

CODING PROGRAMME

CONDITIONAL LOGIC

IF STATEMENTS

BLACK CODHER
CODING PROGRAMME

An `if` statement lets you run a piece of code if an expression is `true`:

```
const iAmAQueen = true;

if (iAmAQueen) {
    console.log('yes, I am a Queen 👸');
}
```

IF STATEMENTS

BLACK CODHER
CODING PROGRAMME

You can also use an expression inside an `if` statement.

```
const coaches = 20;
const students = 24;
const pizzas = 25;

const totalPeople = coaches + students;

if (totalPeople > pizzas) {
  console.log('We have more people than pizzas!');
}

if (students > pizzas) {
  console.log('We have more students than pizzas!');
}
```

IF STATEMENTS

BLACK CODHER
CODING PROGRAMME

You can add an else block to run some code if the expression is `false`.

```
if (totalPeople > pizzas) {
  console.log('We have more people than pizzas.')
} else {
  console.log('We have waaay too much pizza. That can never be bad! :)')
}
```

CONDITIONAL (TERTIARY) OPERATOR

BLACK CODHER
CODING PROGRAMME

The ternary operator is the only JavaScript operator that takes three operands:

- a condition followed by a question mark (`?`),
- then an expression to execute if the condition is truth followed by a colon (`:`),
- and finally the expression to execute if the condition is false. This operator is frequently used as a shortcut for the `if` statement.

CONDITIONAL (TERTIARY) OPERATOR

BLACK CODHER
CODING PROGRAMME

```
function getFee(isMember) {
  return (isMember ? '£2.00' : '£10.00');
}

console.log(getFee(true));
// expected output: "£2.00"

console.log(getFee(false));
// expected output: "£10.00"

console.log(getFee(1));
// expected output: "£2.00"
```

FYI

For the remainder of this course we will using `if`/`else` statements instead of the **tertiary operator** to help cement your understanding of conditional logic.

TASK

BLACK CODHER
CODING PROGRAMME

You are given a variable `marks`. Your task is to print the following grades:

- `A+` if `marks` is greater than `90`.
- `A` if `marks` is greater than `80` and less than or equal to `90`.
- `B` if `marks` is greater than `70` and less than or equal to `80`.
- `C` if `marks` is greater than `60` and less than or equal to `70`.
- `D` if `marks` is greater than `50` and less than or equal to `60`.
- `E` if `marks` is greater than `40` and less than or equal to `50`.
- `F` if `marks` is greater than `30` and less than or equal to `40`.

SWITCH STATEMENTS

A `switch` statement can replace multiple `if` checks and gives a more descriptive way to compare a value with multiple variants.

The `switch` has one or more `case` blocks and an optional default.

SWITCH STATEMENTS

```
switch(x) {
  case 'value1':  // if (x === 'value1')
    ...
    [break]

  case 'value2':  // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}
```

- The value of `x` is checked for a strict equality to the value from the first `case` (that is, `value1`) then to the second (`value2`) and so on.
- If the equality is found, `switch` starts to execute the code starting from the corresponding `case`, until the nearest `break` (or until the end of `switch`).
- If no `case` is matched then the `default` code is executed (if it exists).

SWITCH STATEMENTS

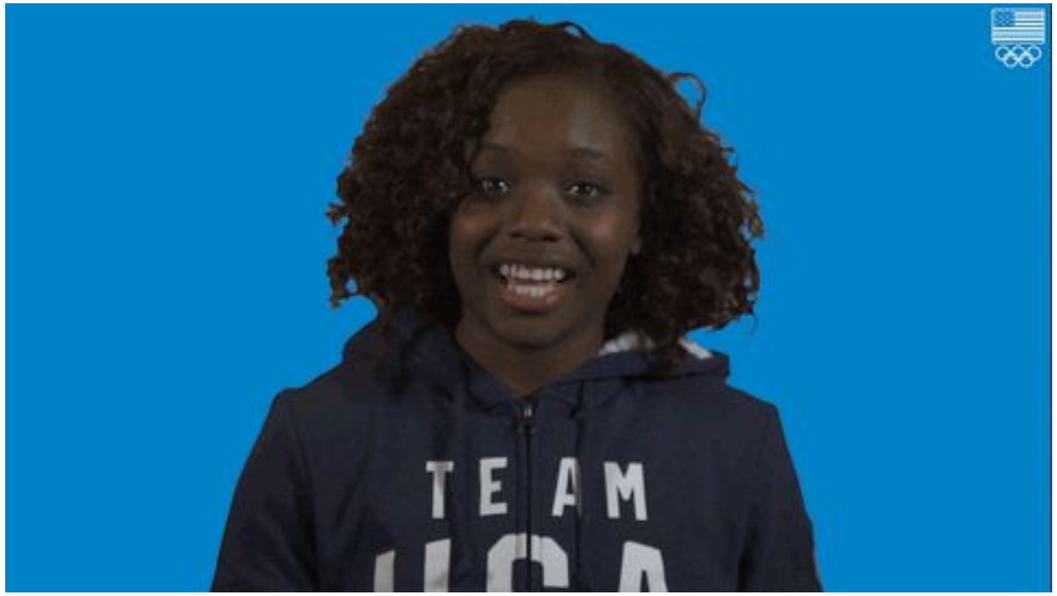
```
let a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Too small' );
    break;
  case 4:
    alert( 'Exactly!' );
    break;
  case 5:
    alert( 'Too large' );
    break;
  default:
    alert( "I don't know such values" );
}
```

Here the `switch` starts to compare `a` from the first `case` variant that is `3`. The match fails.

Then `4`. That's a match, so the execution starts from `case 4` until the nearest `break`.

If there is no break then the execution continues with the next case without any checks.

WHAT YOU'VE LEARNED SO FAR



- What an `if` statement is
- What a `switch` statement is
- When to use a `switch` statement instead of an `if` statement

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

BLACK CODHER

CODING PROGRAMME

SUMMARY

SUMMARY

- We've learnt why a semicolon `;` is important
- What an **object** is
- What a **boolean** is
- How to use maths operators
- What an `if`/`else` statement is
- How to use an `if`/`else` statement
- What `switch` statement is

BLACK CODHER

CODING PROGRAMME

AND FINALLY...

HOMEWORK

BLACK CODHER
CODING PROGRAMME

Rewrite the `if` statements from the previous task as a `switch` statement